Developing IoT Platform with Software as A Service (SaaS) Model for Image Storage System

Dionisius Defilios Hero[1], Nyoman Bogi Aditya Karna[2], Istikmal[3]
[1,2,3]Telecommunication Engineering, School of Electrical Engineering, Telkom University
[1]diondehero@student.telkomuniversity.ac.id, [2]aditya@telkomuniversity.ac.id, [3]istikmal@telkomuniversity.ac.id

***Abstract*—This paper proposes an IoT platform for image storage system using software as a service (SaaS) model. A wireless surveillance system with IoT architecture is more efficient in using storage and uses less bandwidth than a conventional surveillance system. This paper aims to accommodate the specifications for such IoT platform by using the SaaS model. The IoT platform, in the form of a Flask web application framework and MongoDB database server, uses a virtual machine (VM) in a public cloud service to connect to the camera nodes and to store the transmitted images. To fulfill the SaaS model, customized web services and lightweight web-based client applications are developed to manage the camera nodes and display the stored images. This paper proposes an experiment using real-time transmission of sample images sent from a simulated camera node to the IoT platform through a public network. During the experiment, the quality of service of the IoT platform is measured and analyzed using TShark and Free Monitoring by MongoDB to determine whether the IoT platform meets the requirements for an image storage system with SaaS model that can be used for a wireless surveillance system. The proposed IoT platform is able to provide an average delay of 435.3 milliseconds with no packet loss and an average rate of insert and read operations performed of 2.83 operations per second and 2.26 operations per second respectively. Furthermore, the CPU usage of the VM is below 2.5% and the memory usage is below 39% during the experiment.**

I. Introduction

Home surveillance systems have been developed and used extensively in the modern world. Most households use a standard Closed Circuit Television (CCTV) camera surveillance system that captures footage from a camera node and stores the footage in a local file server, which is inefficient in terms of data storage, power inefficiency, and coverage area. With the development of wireless multimedia sensor networks (WMSN) and internet of things (IoT), an IoT-based wireless surveillance system can improve upon the CCTV surveillance system by increasing the coverage area albeit with some problems such as power consumption and signal reliability [1].

Cost-related problems also arise in wireless surveillance systems since the system uses complex image processing methods to conserve power and data bandwidth which requires a separate image processing server [2], a sophisticated camera node [3], or some combinations of the two components [4]. However, the overall cost can be reduced by using a cloud infrastructure while also increasing the scalability and storage of the surveillance system. The cloud service provider can accommodate multiple clients by using the same infrastructure, thus making the service cost cheaper for every client.

To provide users of the IoT platform with further ease of use and reduced cost, the IoT platform is developed using software as a service (SaaS) model. The SaaS model allows the user to use the IoT platform without any further setup and without the burden to develop or modify a separate application to manage the surveillance system. As the SaaS model restricts the user's modification to the IoT platform or the surveillance system, the security and privacy of the data sent by the surveillance system and accessed by the user must be ensured to prevent any security or information breach. The QoS of the IoT platform must also fulfill the specifications for an IoT platform with SaaS model.

A power-efficient wireless surveillance system requires an IoT platform that is efficient and easy to be used by the users. Although the available cloud-based surveillance systems have satisfied the requirements for a smart home surveillance system, they are not effective at using bandwidth and storage space due to the high transmission rate and the number of resources used. Both problems come from the inefficiency of data transmission where live footage is transmitted continuously without any image processing or compression. These problems affect the suitability to implement a power-efficient surveillance system which requires a sufficient IoT platform that uses fewer resources.

This paper proposes an IoT platform for image storage system using the SaaS model that minimizes the resource usage while satisfying the QoS required by the system. This is achieved by developing a customized web application with Flask, customized web services, and by using MongoDB to store images more efficiently. The IoT platform is expected to achieve satisfactory results in terms of network QoS, resource utilization, and database performance. Furthermore, the work done in this paper is expected to inspire more innovations to the field of IoT platforms, wireless surveillance systems, as well as developing SaaS models for other kinds of IoT platforms or technologies. However, this paper is limited by several aspects, which are:

- The IoT platform is developed using the SaaS model in the form of customized web services and a web-based client application that uses Flask web application framework and MongoDB database server.

- The QoS data is taken using TShark and Free Monitoring by MongoDB.
- The unaccounted resources of the VM are used by the kernel and other services.
- Security measures are not developed extensively for the IoT platform.

II. Basic Concepts

A. Wireless Camera Surveillance System

A wireless camera surveillance system is a system of camera nodes that are connected to a file server via a public or private wireless network. The camera nodes in a wireless surveillance system typically consist of a microcontroller with an attached camera or other sensors, a power source, and a wireless adapter module. The camera node captures and sends the raw footage to the file server with minimum processing involved due to the limitation of the processor available. However, because the camera nodes do not necessarily require power cable and data cable installments, the wireless system is more flexible compared to a conventional surveillance system based on the placement of the camera nodes.

Other types of wireless surveillance systems use a dedicated server to process all the images sent by the camera nodes to conserve storage and data transmission of the file server [2] [4], which provides more efficient data consumption as the wireless system transmits more information while using less bandwidth. A complex system or sophisticated image processing method is beneficial if the area needed to be covered by the camera nodes is large or busy with activity. However, for home security, a wireless surveillance system needs to be simple enough to be installed and operated while using the least amount of power and bandwidth as possible.

Several improvements have been made to the design of wireless surveillance systems to reduce the power consumption and bandwidth usage while retaining the same quality and quantity of information from the camera nodes. One example of such improvement is using complex algorithms on camera nodes to process and filter the captured footage before selectively transmitting it to the file server [3]. Another type of algorithm can be applied to the camera node that limits the frame per second (FPS) to a minimum amount when no active movements are observed by the camera node. This new algorithm can reduce the power consumption and bandwidth usage significantly in specific placements such as when the camera node is placed in server rooms or high-value vaults. In these placements, there is no significant movement in a normal situation which requires less transmission rate to provide the same amount of information.

B. IoT Platform for Smart Home System

A smart home system is a part of the many applications of IoT in the field of human lifestyle. The smart home system consists of interconnected IoT devices that are inside and outside of the house to monitor the conditions of the house and control various home appliances to suit the user's needs and preferences. A smart home system is typically provided to the user in the form of an IoT platform that is connected to the IoT devices inside the house. The user can access the IoT platform through a mobile phone application, a web application, or a built-in system that is planted inside the house.

The IoT platform for a smart home system can offer more benefits to the user when it is integrated with public cloud services. These public cloud services offer infrastructure as a service (IaaS) backend that provides a public hosting site and additional processing power for customized applications and services [5]. However, the added benefits also require several considerations before the integration could be implemented. One of the considerations is the data storage for the IoT platform.

Since a smart home system consists of many different IoT devices that transmit and receive different kinds of data, a NoSQL database is the perfect fit to be used by the IoT platform. A commonly used NoSQL database is MongoDB that already has a monitoring application and can store images efficiently alongside other different data types. The usage of MongoDB is especially effective in the aspect of surveillance of a smart home system. A smart home surveillance system typically consists of camera nodes or WMSN that are connected to an IoT platform to detect trespassers or thefts. The camera nodes send a continuous stream of images and metadata to the IoT platform. By using MongoDB, the collection of images can be stored alongside the metadata which requires less time to be found and fetched.

C. Software as A Service Model

Software as a service (SaaS) is a business model that provides solutions in the application layer of the Open System Interconnection (OSI) architecture to a customer as a service.This leads to a more compact approach to solving a customer's needs or problems. The providers of SaaS offer solutions to their customer's needs in the form of software or application. The solutions can be web-based or mobile-based depending on the customer's request. The maintenance or modifications of the proposed application are also determined between the solution provider and the customer to avoid any miss-communication and mishandling of the client application. Furthermore, the backbone cloud infrastructure is inaccessible by the customer and further customization of the provided application or its infrastructure is typically very limited.

Research regarding wireless surveillance systems with SaaS model concludes that using SaaS model in a surveillance system is inefficient compared to other types of XaaS models [6]. A surveillance system with SaaS model, also known as video surveillance as service (VSaaS), requires cameras that are compatible with the VSaaS system. Unfortunately, these cameras needed further configurations and were limited in the market. There is also concern with the footage storage since a wireless surveillance system requires a massive amount of storage. However, customizable small cameras nowadays are more affordable than in the past and the cameras are now easier to be configured which eliminates the concern for lack of compatible cameras.

For the cloud infrastructure, the solution providers of the VSaaS model need to use platform as a service (PaaS) or infrastructure as a service (IaaS) solutions available in the current market to reduce development and operating costs. By taking advantage of the cloud infrastructure provided by the PaaS or IaaS providers, the VSaaS solution providers only need to develop the custom web services and the client application to meet their customer's demands.

D. QoS Parameters

QoS is defined as a mechanism or a method to determine whether a service or a system in a network can fulfill its objective based on its service characteristics. To fulfill the QoS for IoT platform with the SaaS model for image storage system in wireless surveillance system, the images transmitted by the camera nodes must be viewed in real-time and not corrupted, while the IoT platform is efficient in using storage space and computational power. Several analyses such as network QoS analysis, resource utilization analysis, and database performance analysis could be used to determine whether the IoT platform fulfills the described QoS.

1) Network QoS Parameters: The network QoS parameters consist of delay, jitter, throughput, and packet loss. Delay is the amount of time needed for a packet to be transmitted to the receiver, while jitter is the variance in delay. Throughput is the rate of data transferred to the receiver, while packet loss is the percentage of packets that are lost compared to the total packets transmitted. The threshold for each parameter to be considered satisfactory is dependent on the type of service provided.

In the case of still images in a surveillance system, there is an emphasis on the low delay of each packet received without a packet loss [7]. This is because the images must be as close to real-time as possible and every image contains valuable information, especially in the proposed IoT platform because of the low FPS of the camera nodes. Meanwhile, throughput can be measured to determine the amount of traffic received by the surveillance system, and delay variance is used to determine the variability of the packet's arrival time.

2) Resource Utilization Parameters: The resource utilization analysis uses CPU usage and memory usage data to determine whether an application is using the available computing and memory resources efficiently. The measure of resource utilization is the percentage of time that a resource is being used compared to the time the resource is available. When a resource becomes overused or highly used compared to other resources, that resource becomes critical. A critical resource indicates an inefficient design of the system or the system is having a peak load.

Various processes use the CPU to perform calculations and execute certain instructions. The more calculations and instructions that the CPU executes in a time period, the higher the throughput of the CPU. Thus, high CPU utilization does not always indicate a performance problem when the CPU throughput is also high. However, when the CPU utilization is high but the throughput is low, there is a process that uses the CPU inefficiently.

Memory usage is determined by the activity and usage of small components that make up the memory called pages. Processes use these pages by locating unused pages and store them with temporary data. A page scan is a process where a CPU cycle is used to select unused pages for other processes. A high rate of page scans indicates that poor memory utilization is becoming a problem.

3) Database Performance Parameters: Database performance analysis measures the throughput and response time of the database in completing operations. Both parameters depend on the specifications of the system such as processing power and the type of data being processed. Database throughput determines the overall performance of the database by measuring the number of operations completed in a given time, while database response time determines the performance of a single query or operation by measuring the service time of each query of operation.

Database throughput is analyzed through different scenarios to determine the database performance in handling simple operations, complex operations, and a large number of operations. Depending on the service type of the system, some scenarios are more likely to provide appropriate information than other scenarios.

III. Proposed IoT Platform

According to [8], the framework for a cloud-based video surveillance system consists of three parts:

- The public cloud server which consists of a web server, a media server, and a database server. The web server is used for storing the website while the media server is used to store the footage sent by the camera nodes. The metadata and user information is stored in the database server.
- The web services which are used to act as the middleware between the cloud servers and client application to control the camera nodes and manage user information.
- The web-based client application which is used as the interface for the user to control the camera nodes and view the stored footage.

In the proposed model of the IoT platform, this paper uses a virtual machine (VM) provided by a hosting service to act as the public cloud server to make storage more efficient.

As shown in Figure 1, the connection between the client, the camera node, and the public cloud server is available through the internet. The client access the IoT platform through the web-based client application which directly connects to the web server that acts as the cloud manager of the IoT platform. The web server, database server, and the media server is contained in the same VM to improve efficiency in the cost of negligible CPU and memory usage. The camera node only transmits images directly to the public cloud server for storage, thus the client can only access the images through the IoT platform.

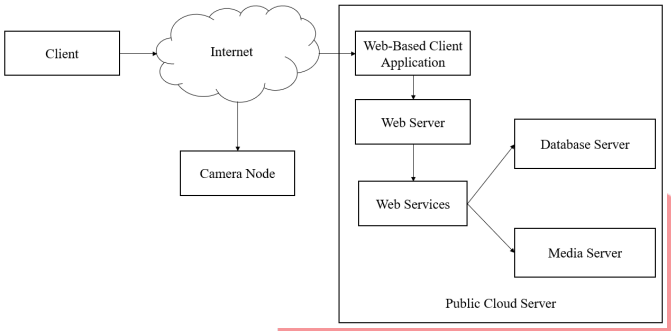

Fig. 1. The proposed system design of IoT platform for a power-efficient smart home surveillance system.

TABLE I
THE SPECIFICATIONS OF THE PROPOSED IOT PLATFORM FOR THIS PAPER.

Type	Specification
Public Cloud Service	A2Hosting
Type of Cloud Service	Unmanaged VPS
VM Operating System	Ubuntu Server 16.04.6 LTS
Web Application Framework	Flask
Web Server	Gunicorn
Proxy Server	Nginx
Database Server	MongoDB
VM RAM	512 MB
VM Storage Space	20 GB
Bandwidth	2 TB

IV. SYSTEM SPECIFICATIONS

As shown in Table I, this paper uses A2Hosting public cloud services to provide a virtual private server (VPS) designated as the server where the proposed IoT platform is installed. The IoT platform itself is a web-based application made with Flask and written in Python. MongoDB is used as the database server because of its NoSQL architecture. The combination of Gunicorn and Nginx is used as the web server and proxy server respectively to connect the web application to the internet. As for the OS and hardware specification, the VPS uses Ubuntu Server 16.04.6 LTS as the kernel of the virtual machine with 512 MB of RAM, 20 GB of storage, 2 TB of provided bandwidth, and Intel(R) Xeon(R) CPU E5-2620 v3 @2.40GHz (with SSE4.2) as the processor.

V. EXPERIMENT SETUP

To fulfill the QoS for IoT platform with the SaaS model for image storage system in a cloud-based surveillance system, the images transmitted by the simulated camera nodes must be in real-time and not corrupted while being efficient in storage usage and computational power. As such, the analyzed network parameters from the experiment are the total delay and the throughput related to the transmitted sample images. Moreover, the analyzed hardware parameters from the experiment are the CPU and memory usage. Lastly, the database performance parameters are the rate of insert operations and read operations. As shown in Table II, these parameters are measured using TShark as the network analyzer tool and Free Monitoring by MongoDB as the resource monitoring and database monitoring tool.

TABLE II
THE PARAMETERS THAT WILL BE MONITORED DURING IMAGE TRANSMISSIONS TO THE IOT PLATFORM.

Analysis Category	Parameter	Tool of Measurement
Network QoS	Delay	TShark
	Throughput	TShark
Resource Utilization	CPU Usage	Free Monitoring by MongoDB
	Memory Usage	Free Monitoring by MongoDB
Database Performance	Rate of Insert Operations	Free Monitoring by MongoDB
	Rate of Read Operations	Free Monitoring by MongoDB

Due to the unavailability of a power-efficient camera node, this paper could only uses Postman as simulated camera nodes. The Postman simulates a camera node by transmitting sample images to the IoT platform. The network QoS parameters, the resource utilization parameters, and the database performance parameters are monitored and analyzed during the transmission of sample images using TShark as a network analyzer tool, while the resource utilization and database performance are monitored using Free Monitoring by MongoDB.

The image transmission experiment is done three times to notice the effect of random internet traffic to the packets received by the IoT platform. Each trial has different simulated camera nodes and user accounts to simulate different sources of transmission. After each trial, user queries are done to check the images received by the IoT platform. After all three trials are done, the measured parameters are analyzed and compared with each other. The analysis results are then concluded to finish this paper.

VI. EXPERIMENT ANALYSIS

A. Network QoS Analysis

This paper uses the methods described in Chapter 5 to determine whether the proposed IoT platform has achieved QoS of SaaS standards. Fig. 2 shows the average delay of image packets in each experiment trial. Although the internet capability of the simulated camera node is not ideal, the situation may serve as an approximation for the real working condition when the camera nodes and the IoT platform are implemented correctly. The average delay is calculated by analyzing the HTTP packets captured by TShark when the images are transmitted from simulated the camera nodes.

A total of three trials are conducted to account for random internet traffic conditions. Each trial sends a stream of 240 image packets in the span of 120 seconds with 500 milliseconds interval to a different account created for the experiment. The second trial is shown to have the lowest variance in delay and the third trial is shown to have the highest variance in delay. The high delay variance in the third trial is assumed to be due to the poor internet connection used by the simulated camera node.

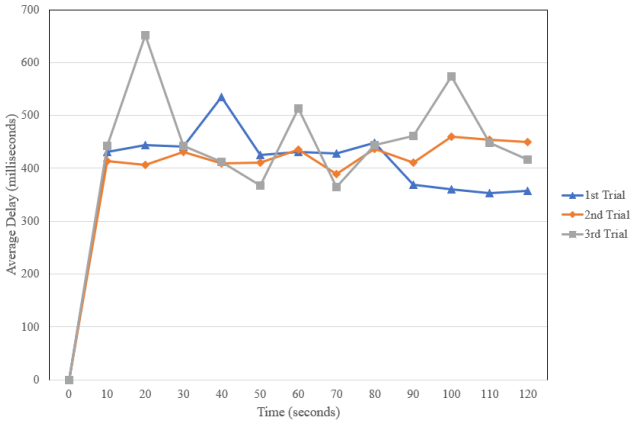

Fig. 2. The average delay from the measured packets received by the IoT platform.

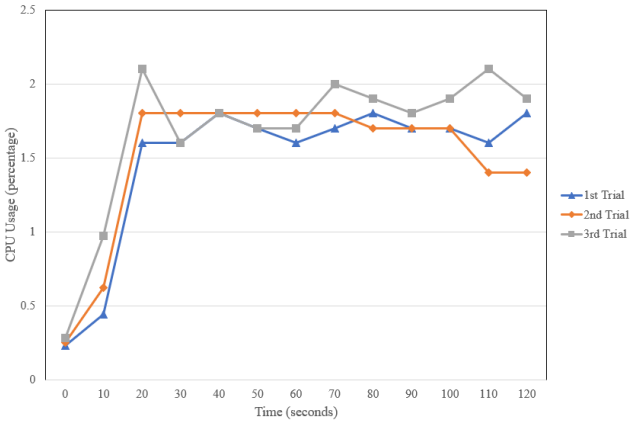

Fig. 3. The CPU usage percentage during image transmission.

B. Resource Utilization Analysis

The resource utilization analysis describes how effective are the CPU and memory usage of the VM when the IoT platform receives streams of images. The CPU usage is measured during the image transmission experiment with the result as shown in Fig. 3. All trials are conducted while the CPU usage of the system was initially at around 0.2% and the CPU usage increases until 1.75% on average. After 120 seconds, the CPU usage drastically decreases to the initial value of just above 0.2% across all trials. The CPU usage is not affected significantly because the database only creates new documents for each image to be stored in. Since the writing process does not require scanning the whole collection of documents, the requests to make new documents are handled without using the CPU too much.

Meanwhile, Fig. 4 shows that memory usage is affected more significantly than CPU usage during the experiment. Each trial has an average memory usage increase by 0.5% due to how MongoDB stores image files to the documents using GridFS. The increases in memory usage between trials are due to additional memory used by MongoDB for query purposes

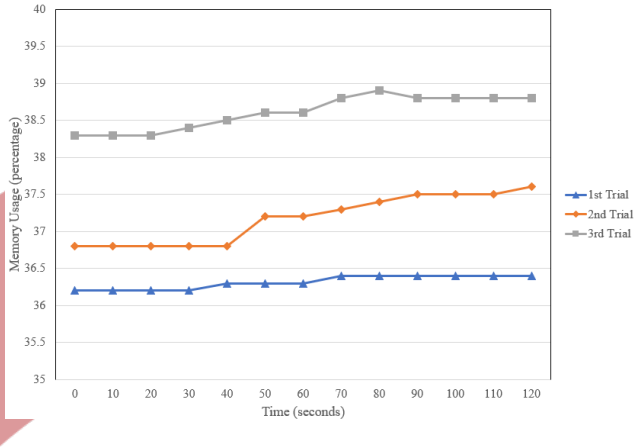

Fig. 4. The memory usage percentage during image transmission.

TABLE III
DATABASE PERFORMANCE RESULTS DURING IMAGE TRANSMISSION.

Parameters	1st Trial	2nd Trial	3rd Trial
Documents updated	240	240	240
Average write operations execution times	2.78 op/second	2.89 op/second	2.82 op/second
Uptime of the IoT Platform	100%	100%	100%

outside the image transmission experiments. The flaw in the IoT platform database design and the limited memory that is provided by the hosting service also contributes to the high usage of memory.

C. Database Performance Analysis

This paper uses Free Monitoring by MongoDB to record database performances. Database performances are indicated by the rate of insert operations and read operations performed during a time period. The Table III shows that the average rates of insert operations performed during the experiment are close to three operations per second. By having a fast rate of insert operations, the IoT platform does not have additional delays when receiving streams of images which enables the IoT platform to lessen the impact of heavy traffic. The fast rate of insert operations is due to the low size of each document that also has simple relationships with other types of collections.

Table IV shows that all of the user queries are completed with low execution times. This is due to the low size of each document and the limited attributes that one user query could have. Also, not all information is retrieved from the database to save time for each user query. In the case of this experiment, only the camera node source is used as the basis of the queries sorted by the date of the upload. With the short read operation execution time, the IoT platform can handle many image requests by various users even during heavy traffic. Thus, users do not experience any additional delay while using the IoT platform.

VII. CONCLUSION

The working process of this paper is concluded in this section.

TABLE IV
DATABASE PERFORMANCE RESULTS DURING USER QUERIES.

Parameters	1st Trial	2nd Trial	3rd Trial
Documents updated	240	240	240
Average read operations execution times	2.23 op/second	2.29 op/second	2.27 op/second
Uptime of the IoT Platform	100%	100%	100%

1) This paper proposed an IoT platform for image storage system using the SaaS model. Flask is used to build the web application paired with MongoDB as the database server. Meanwhile, the role of the web server is handled by Gunicorn and Nginx. This paper rented a shared VPS to achieve less cost to deploy the IoT platform while still being able to be accessed publicly. The monitored QoS parameters as well as the database performance parameters show that the IoT platform can meet the requirements for an online surveillance system.
2) The IoT platform is able to operate with an average delay of 453.3 milliseconds with no packet loss. From the monitoring results of the experiment, all packets are received by the IoT platform with high QoS standards and there is no significant burden to the VPS as well as to the database application.
3) During the experiment, the CPU usage and the memory usage of the VM are below 2.5% and below 39% respectively. The resource utilization analysis shows that the CPU usage is low during the experiment because of the simple write operation and low document size. Meanwhile, the RAM usage is affected significantly due to the GridFS process that is used by the MongoDB to store images.
4) MongoDB is observed to have an average rate of insert and read operations performed of 2.83 operations per second and 2.26 operations per second respectively. The database performance analysis shows that the IoT platform has a fast rate of insert operation due to the simple design of each document and few processes are needed to be added to the database.

VIII. SUGGESTION

These suggestions are given to help the readers acknowledge what this paper lacks so improvements can be made for future research.

- The proposed IoT platform is designed on a specific algorithm used by the camera node, thus requiring customization on the camera nodes to be compatible with the proposed IoT platform. A different SaaS model or a different design of the IoT platform can be developed to solve the customization issue.
- Different types of XaaS can be developed to improve the network QoS parameters, resource utilization parameters, and database performance parameters measured in this paper.
- Different database application and database models can improve the database performance results by changing how the IoT platform stores and retrieve images in the database.

REFERENCES

[1] Y. Ye, S. Ci, A. K. Katsaggelos, Y. Liu, and Y. Qian, "Wireless video surveillance: A survey," *IEEE Access*, vol. 1, pp. 646–660, 2013.
[2] Q.-A. Kester, O. Willis Chibueze, and M. Dolapo Asisat, "A surveillance wireless camera sensor network system for intrusion detection using image processing techniques," *Journal of Engineering and Applied Sciences*, vol. 10, Jun 2015.
[3] J. P. Singh, M. K. Mishra, and M. A. Khan, "Energy-efficient approach towards video-based sensor networks (wireless) beneath barrier coverage," in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Jul 2017, pp. 1–5.
[4] D. Wu, S. Ci, H. Luo, Y. Ye, and H. Wang, "Video surveillance over wireless sensor and actuator networks using active cameras," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2467–2472, Oct 2011.
[5] A. Sarhan, *Cloud-Based IoT Platform*, 01 2019, pp. 116–147.
[6] D. Neal and S. M. Rahman, "Video surveillance in the cloud-computing?" in *2012 7th International Conference on Electrical and Computer Engineering*, 2012, pp. 58–61.
[7] ITU-T, *G.1010: End-user multimedia QoS categories*, Nov 2001.
[8] M. A. Hossain, "Analyzing the suitability of cloud-based multimedia surveillance systems," in *2013 IEEE 10th International Conference on High Performance Computing and Communications 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, Nov 2013, pp. 644–650.